Tyale Schneider Rodrigues

Um Sistema Tolerante a Intrusão Baseado em Técnicas de Diversidade de Software e Virtualização

São José – SC

agosto / 2010

Tyale Schneider Rodrigues

Um Sistema Tolerante a Intrusão Baseado em Técnicas de Diversidade de Software e Virtualização

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Emerson Ribeiro de Mello, Dr.

Curso Superior de Tecnologia em Sistemas de Telecomunicações
Instituto Federal de Santa Catarina

São José – SC

agosto / 2010

Monografia sob o título *"Um Sistema Tolerante a Intrusão Baseado em Técnicas de Diversidade de Software e Virtualização"*, defendida por Tyale Schneider Rodrigues e aprovada em 02 de agosto de 2010, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Emerson Ribeiro de Mello, Dr.
Orientador

Prof. Tiago Semprebom, M. Eng.
IFSC

Prof. Ederson Torresini, M. Sc.
IFSC

Você pode viver até os cem anos se abandonar todas as coisas
que fazem com que você queira viver até os cem anos.
Woody Allen

Agradecimentos

Agradeço a todos aqueles que de alguma forma participaram do processo de desenvolvimento deste trabalho.

Resumo

Muitos estudos para o desenvolvimento de soluções de segurança em ambientes computacionais têm sido realizados nos últimos tempos. A cada dia também, junto com o desenvolvimento de soluções, são descobertas novas vulnerabilidades de *softwares*. Nenhum *software* está imune a vulnerabilidades. Desta forma é cada vez mais difícil conseguir prevenir-se de ataques de pessoas mal intencionadas, que buscam roubar informações ou simplesmente corromper os dados. Como o desenvolvimento de ambientes totalmente imunes a invasões é extremamente complexo e muitas vezes inviável financeiramente, a solução viável é o desenvolvimento de ferramentas que ajudem a minimizar ao máximo os danos causados em caso de invasão.

Sendo assim o presente trabalho apresenta um sistema tolerante a intrusões baseado em técnicas de diversidade de *software* e virtualização. Este sistema funciona em conjunto com sistemas de detecção de intrusão (*Intrusion Detection System*(IDS)), que identificam ataques e geram alertas sobre estes eventos. Com base nesses alertas o sistema executa diversas ações para minimizar os danos causados pela invasão e garantir que o sistema volte a um estado seguro.

No desenvolvimento do trabalho buscou-se utilizar soluções *open source* para manter baixo o custo de desenvolvimento. Também buscou-se simplificar a implementação do sistema a fim de reduzir as falhas de configuração e de desenvolvimento.

Palavras chave: segurança computacional, virtualização, tolerância a intrusão.

Abstract

Many studies to develop security solutions in computing environments have been conducted recently. Every day, along with development of these solutions, new software vulnerabilities are discovered. No software is immune to these vulnerabilities. Thus it is increasingly difficult to protect computing environments from attacks by malicious people, seeking to steal information or simply corrupt data. Since the development of environments totally immune to invasions is not possible, the feasible solution is the development of tools that helps to minimize at most the damage in case of invasion.

Therefore this paper presents an intrusion tolerant system based on techniques of diversity and software virtualization. This system works with intrusion detection systems (IDS) that identify attacks and generate alerts about these events. Based on this alert, the system performs several actions to minimize damage in case of invasion and ensure that the system remains secure.

In developing this work we attempted to use open source solutions to keep the cost of delelopment down. Also sought to simplify implementation of the system to reduce the risks of misconfiguration and development mistakes.

Keywords: computer security, virtualization, intrusion tolerance.

Sumário

Lista de Figuras

1 Introdução p. 10

1.1 Motivação . . . . . p. 10

1.2 Objetivos do trabalho . . . . . p. 11

1.3 Organização do texto . . . . . p. 12

2 Fundamentação teórica p. 13

2.1 Tolerância a intrusão . . . . . p. 13

2.2 Sistemas de detecção de intrusão . . . . . p. 14

2.2.1 Tipos de IDS . . . . . p. 15

2.3 Virtualização . . . . . p. 17

2.3.1 O monitor de máquinas virtuais . . . . . p. 18

2.3.2 Tipos de virtualização . . . . . p. 19

2.3.3 Ferramentas de virtualização . . . . . p. 20

2.3.3.1 *VirtualBox* . . . . . p. 20

2.3.3.2 *Xen* . . . . . p. 21

2.4 Diversidade de *software* . . . . . p. 22

3 Um sistema tolerante a intrusão p. 23

4 Implementação do protótipo p. 28

4.1 Configurações básicas do sistema . . . . . p. 28

4.2 Implementação das VMs . . . . . p. 29

4.3 Configuração do *OSSEC* Host Intrusion Detection System (HIDS) . . . p. 29

4.3.1 Configuração dos agentes *OSSEC* . . . . . p. 29

4.3.2 Configuração do servidor *OSSEC* . . . . . p. 30

4.4 Implementação do SDIR . . . . . p. 31

4.5 Implementação da base de dados . . . . . p. 35

5 Experimentos e resultados p. 37

5.1 Experimetos . . . . . p. 38

6 Conclusões p. 41

6.1 Trabalhos futuros . . . . . p. 41

Lista de Abreviaturas p. 43

Referências p. 45

Lista de Figuras

1 Exemplo de sistema com HIDS . . . . . p. 15

2 Exemplo de sistema com Network Intrusion Detection System (NIDS) . p. 17

3 Implementação da virtualização total . . . . . p. 19

4 Implementação da paravirtualização . . . . . p. 20

5 Sistema proposto . . . . . p. 25

6 Estrutura utilizada para habilitar a resposta ativa no arquivo `ossec.conf` p. 30

7 Estrutura utilizada para criar um comando no arquivo `ossec.conf` . . p. 31

8 Exemplo de arquivo utilizado como base de dados . . . . . p. 35

9 Cenário simulado nos experimentos . . . . . p. 38

10 Desempenho do sistema nos dois experimentos . . . . . p. 39

11 Desempenho do sistema nos dois experimentos . . . . . p. 39

1 *Introdução*

Este capítulo apresenta, nas próximas seções, as principais motivações para realização deste trabalho, bem como os objetivos principais e secundários que se buscou alcançar. Apresenta ainda na Seção 1.3 a organização do texto nos próximos capítulos.

1.1 Motivação

Ambientes computacionais desempenham, cada vez mais, um papel muito importante em nossa sociedade. A demanda por conectividade e segurança para esses ambientes aumenta diariamente, assim como a nossa dependência por esses sistemas. Tendo em vista a relevância desses ambientes para o funcionamento, tanto para serviços básicos de nosso cotidiano como para as empresas, é preciso que esses operem corretamente ou recuperem-se rapidamente, mesmo quando apresentam falhas. Grande parte dessas falhas acontece devido a ataques bem sucedido (LUNT, 1993). Estes ataques são possíveis pois exploram vulnerabilidades de *softwares* (muitas vezes desatualizados) e má configuração das ferramentas de segurança. Após a ocorrência de um ataque, independentemente do motivo, é necessário iniciar um procedimento de recuperação desse sistema. Inicialmente o sistema é desligado e deve ser encaminhado para a perícia. O papel da perícia é tentar rastrear o ataque e identificar o que exatamente foi acessado ou destruído. Após a perícia, devemos formatar o disco rígido, reinstalar o sistema e configurá-lo novamente. Todo esse procedimento exige tempo e na maioria das vezes perda de tempo significa prejuízo. Deve-se levar também em consideração que dependendo do tipo de ataque, pode ser mais fácil ou mais complicado identificá-lo. Quando um site é invadido e pichado identificamos rapidamente essa invasão. Isso porque o objetivo nesse caso era ser notado. Uma máquina pode ser atacada para ser utilizada como zumbi e, por exemplo, ficar disparando /textitspams para listas de *e-mails*. Em outro caso o ataque pode ocorrer com objetivo de roubar informações confidenciais. Em ambos os casos é objetivo não ser notado. O tempo gasto para detecção de um ataque é variável, dependendo do administrador identificar

esse ataque. Para garantir que os sistemas computacionais permaneçam disponíveis (ou voltem a funcionar rapidamente) e seguros mesmo quando atacados, é necessário o desenvolvimento de ferramentas que viabilizem a tolerância a intrusões (AMOROSO, 1999). O desenvolvimento dessas ferramentas tem sido alvo de pesquisas há muitos anos. Porém, grande parte das soluções desenvolvidas não são viáveis financeiramente ou em termos de complexidade de implementação.

Recentemente, tecnologias de virtualização têm sido utilizadas para o desenvolvimento de sistemas tolerantes a intrusão (LAUREANO et al., 2003). Dessa forma, várias réplicas do servidor são configuradas em diferentes ambientes virtualizados. Todas as réplicas são executadas em uma mesma máquina real. Dessa forma mesmo que uma das réplicas seja atacada, as outras réplicas estarão funcionando de maneira correta. A grande vantagem da utilização dessa técnica é evidentemente a redução do custo de implementação das réplicas, já que somente um computador é utilizado . A desvantagem do uso das réplicas é que uma falha explorada em uma réplica pode ser explorada em todas as outras. Uma técnica muito utilizada para sanar este problema é a da diversidade de *software*. Esta técnica garante que as vulnerabilidades encontradas em um servidor não sejam encontradas em outro (AVIZIENIS, 1984).

1.2 Objetivos do trabalho

O principal objetivo deste trabalho consiste na concepção de um sistema tolerante a intrusão aliado a técnicas de diversidade de *software* e virtualização. Objetiva-se um sistema reativo que ao detectar uma invasão, garanta que o serviço provido ficará indisponível o menor tempo possível, até sua reconfiguração, através de uma nova réplica. Tal reconfiguração será baseada em técnicas de diversidade de *software*, visando assim subir uma pilha de *software* diferente daquela que fora invadida, esperando assim que essa nova réplica não possua as mesmas vulnerabilidades apresentadas pela réplica que sofreu a invasão. Sistemas de detecção a intrusão (textitIntrusion Detection System (IDS)) serão usados nas réplicas (máquinas virtuais) e estes irão interagir com o Intrusion Detection System (IDS) da máquina hospedeira. O sistema a ser desenvolvido fará uso dos eventos gerados pelo IDS para que então possa tornar suas decisões, por exemplo, remover uma réplica que fora invadida e carregar outra réplica com uma pilha de *software* diferente.

Como objetivo secundário busca-se um sistema relativamente simples, a fim de diminuir a probabilidade de falhas na implementação e configuração. É também objetivo secundário

desenvolver um sistema com baixo custo de implementação, através do uso de somente uma máquina física hospedando diversos servidores virtuais.

1.3 Organização do texto

O trabalho está organizado da seguinte forma: no Capítulo 2 é apresentada a fundamentação teórica necessária para o desenvolvimento deste trabalho. No Capítulo 3 o sistema proposto é apresentado em detalhes, bem como seu funcionamento. No Capítulo 4 é apresentada a implementação do sistema, os testes realizados e os resultados dos testes. Por fim, no Capítulo 5 são apresentadas as conclusões sobre este trabalho.

2 *Fundamentação teórica*

Este capítulo apresenta a fundamentação básica teórica necessária para o desenvolvimento do sistema proposto. São abordados temas referentes a tolerância a intrusões, sistemas de detecção de intrusão, virtualização e diversidade de projeto.

2.1 Tolerância a intrusão

Nenhum sistema computacional é totalmente imune a intrusão. Desta forma um sistema tolerante a intrusões é um sistema que mesmo após um ataque, permanece seguro e livre de erros (VERISSIMO, 2003). Segundo (BESSANI et al., 2008), um sistema tolerante a intrusões é um sistema capaz de disponibilizar um serviço seguro e estável de forma contínua independentemente de intrusões em um determinado número de seus componentes. Sendo assim as principais características do conceito de tolerância a intrusões são manter o sistema seguro e funcionando corretamente, mesmo que haja diminuição no desempenho do sistema.

Para que tais características sejam conseguidas é necessário garantir que o sistema não apresente nenhum componente que, se atacado, provoque falha em todo sistema. A maneira mais fácil de garantir isto é distribuíndo ou replicando os dados do sistema (CORREIA et al., 2001). Desta forma a falha em um determinado ponto torna-se independente de outra falha.

Para descrever a manifestação de intrusões em sistemas computacionais, (VERISSIMO, 2003) apresentou o modelo Atack - Vulnerability - Intrusion (AVI), que descreve uma intrusão como um ataque bem sucedido ao explorar uma vulnerabilidade. É importante perceber que todas estas ações (ataques, vulnerabilidades e intrusões) são falhas em diferentes níveis. A vulnerabilidade é uma falha no sistema específico, seja na sua implementação ou na sua configuração. Já um ataque caracteriza-se como uma ação que busca explorar vulnerabilidades de maneira maliciosa. Por fim, a intrusão é o resultado

de um ataque bem sucedido. A descrição acima apresentada auxilia o entendimento do processo de intrusão bem como as técnicas que podem ser utilizadas para prevenção.

2.2 Sistemas de detecção de intrusão

Sistemas de Detecção de Intrusão (IDS) são dispositivos ou *softwares* que monitoram a rede ou o sistema com intuito de identificar atividades maliciosas ou violações de políticas e em seguida gerar relatórios para uma estação de gerenciamento (AMOROSO, 1999).

A principal função dos IDSs é a identificação de possíveis incidentes, registrando informações na tentativa de pará-los e denunciá-los aos administradores da rede (ULVILA and GAFFNEY, 2003). Além disso, as organizações utilizam IDSs para outros fins, tais como para identificação de problemas com as políticas de segurança. Dessa forma um IDS torna-se um complemento necessário à infraestrutura de segurança de quase todas as organizações.

Os IDSs registram informações relacionadas a eventos observados, notificam os administradores da rede de eventos importantes e produzem relatórios. Muitos IDSs podem também responder a ameaças detectadas tentando impedí-las de ter sucesso usando técnicas de resposta ativa. Sendo assim, o grande diferencial do uso de IDSs é que este auxilia o administrador na identificação de intrusões.

Porém existem algumas desvantagens no uso de IDSs. A principal delas é que nem sempre um IDS identifica de maneira correta uma intrusão. Segundo (LUNT, 1993), um IDS pode gerar identificações de quatro tipos:

- Falsos positivos – o IDS identifica um acesso legítimo como sendo um ataque e gera um alerta;
- Falsos negativos – o IDS identifica um ataque como sendo um acesso legítimo e não gera um alerta;
- Positivo real – o IDS identifica corretamente um ataque e gera um alerta;
- Negativo real – o IDS identifica corretamente um acesso legítimo e não gera um alerta.

Para identificar estes eventos os IDSs utilizam basicamente duas técnicas: IDSs baseados em conhecimento e IDSs baseados em comportamento (LUNT, 1993). Quando o

IDS baseia-se em conhecimento, uma base de dados é utilizada. Essa base de dados contém assinaturas de ataques conhecidos pelo sistema. Desta forma o IDS baseado no conhecimento basicamente compara as informações e gera um alerta se a comparação é verdadeira. Já quando o IDS baseia-se em comportamento, ele busca identificar desvios quanto a padrões de normalidade, como, por exemplo, um aumento súbito de tráfego de dados.

2.2.1 Tipos de IDS

Existem basicamente dois tipos de IDS. Os IDSs de máquina (HIDS) e os IDSs de rede (NIDS). Existem ainda IDSs que utilizam técnicas híbridas, explorando as vantagens tanto dos HIDS quanto dos NIDS.

Os HIDS analisam sinais de intrusão na máquina onde estão instalados. Este tipo de IDS utiliza mecanismos de *log* do sistema operacional para identificar acessos e alterações em arquivos de sistema, modificações em privilégios de usuários, processos do sistema e programas em execução (MELL et al., 2003). A Figura 1 mostra o exemplo de um sistema com HIDS. Nessa figura, cada um dos servidores possui um agente HIDS instalado para monitoramento local. Também é possível visualizar um gerenciador HIDS que recebe os alertas dos agentes.

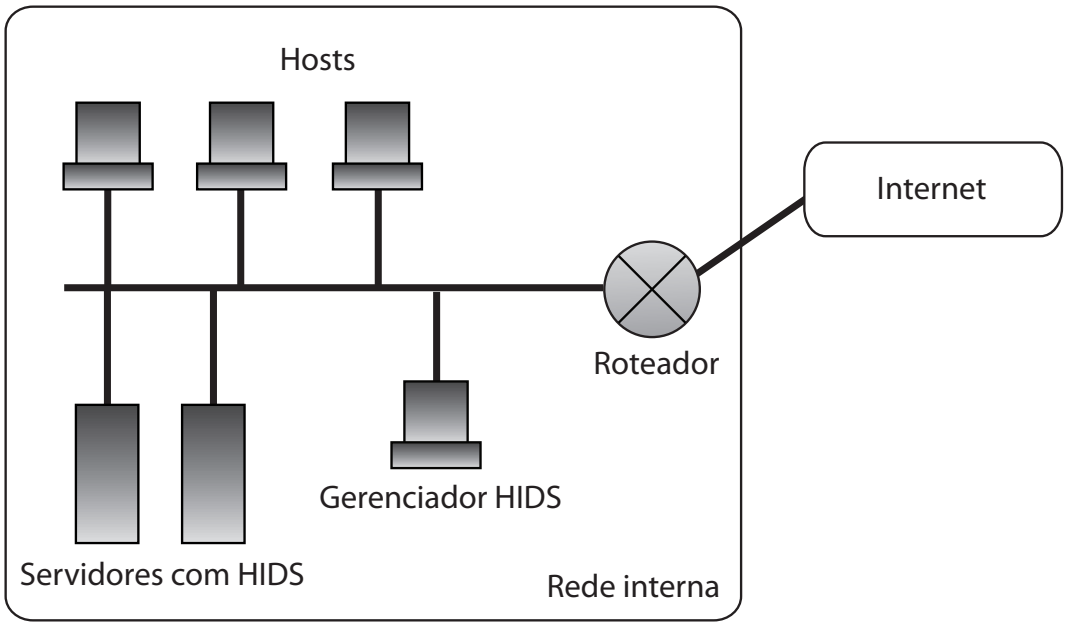

Figura 1: Exemplo de sistema com HIDS

As principais vantagens do uso dos IDSs de máquina são: por basear-se em *logs* do sistema, pode muitas vezes identificar exatamente o que foi feito pelo invasor; geram

menos falsos positivos que os NIDS; sua configuração é mais simples que a do NIDS; é mais difícil de burlar que o NIDS. Já entre as desvantagens do uso deste IDS podemos citar: dependem da capacidade do sistema onde estão instalados; são mais caros; monitoramento focado na máquina e não no ambiente como um todo. Como exemplos de HIDS podemos citar o *OSSEC*[1], o *Osiris*[2] e o *AIDE*[3].

O *OSSEC* é um HIDS gratuito e de código aberto. O *OSSEC* realiza análise de *log*, verificação de integridade, monitoramento de registro, detecção de *rootkits* (um tipo específico de *malware*) e implementação de resposta ativa. O *OSSEC* possui um sistema centralizado e uma arquitetura multi-plataforma, que permite múltiplos sistemas serem monitorados e controlados pelo sistema central. O *Osiris* é também um HIDS gratuito e de código aberto. Implementa a maioria das funcionalidades do *OSSEC*, com exceção da possibilidade de resposta ativa. Também possui arquitetura multi-plataforma, o que permite a implementação de múltiplos sistemas. Por fim o *AIDE*, também código aberto, implementa somente as funções básicas de verificação de *log*. Não possui arquitetura multi-plataforma, sendo indicado para aplicações bem simples.

Os NIDS monitoram o tráfego em um segmento de rede e detectam intrusões analisando os cabeçalhos e conteúdos dos pacotes e identificando *interfaces* de rede atuando de maneira promiscua. Os NIDS são constituidos geralmente dois componentes: os sensores e o gerenciador.

Os sensores são configurados em segmentos de rede distintos que se deseja monitorar e são responsáveis pela captura, formatação e análise dos dados trafegados neste segmento.

O gerenciador é responsável por receber os alertas gerados nos sensores. Ele gerencia de modo integrado esses alertas e define os tipos de resposta para cada comportamento suspeito detectado (PAXSON, 1998). Toda a comunicação entre sensores e o gerenciador acontece de maneira criptografada, a fim de aumentar o nível de segurança da ferramenta. A Figura 2 mostra um sistema com NIDS. Nessa figura percebemos que não há agentes IDSs instalados nos servidores, mas sim em *enlaces* distintos da rede. Assim como no HIDS, aqui podemos visualizar um gerenciador NIDS que recebe os alertas gerados pelos agentes.

Entre as principais características positivas destacam-se: monitoramento de portas da rede; não afeta diretamente o sistema onde é instalado; visão da rede como um todo.

[1] `http://www.ossec.net`
[2] `http://osiris.shmoo.com/`
[3] `http://www.cs.tut.fi/~rammer/aide.html`

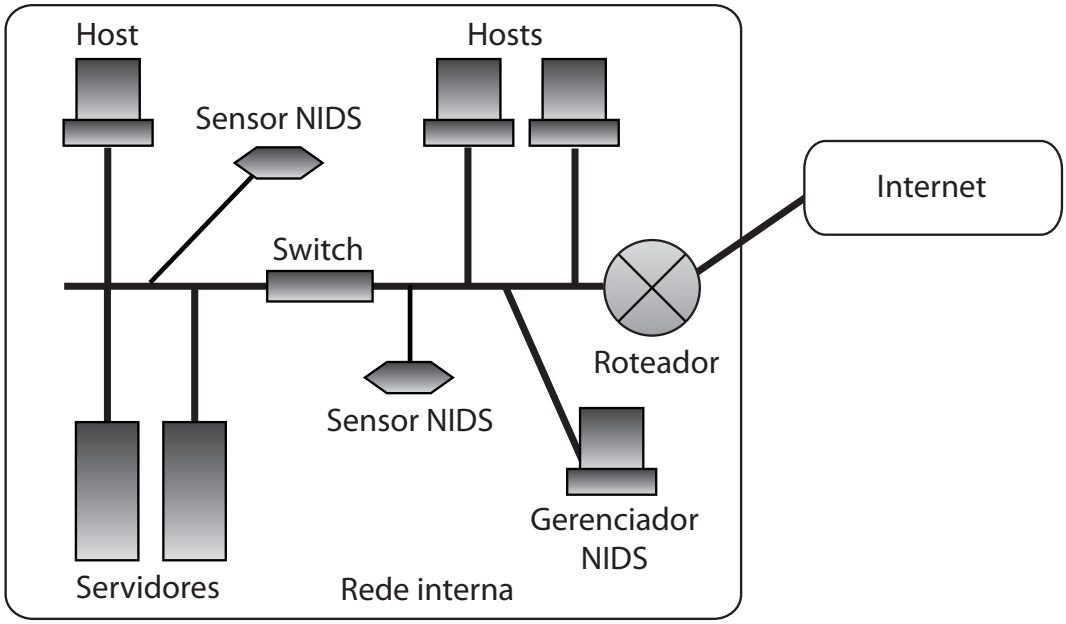

Figura 2: Exemplo de sistema com NIDS

Já como desvantagens podemos citar: não monitoram tráfego criptografado; pode sobrecarregar o tráfego da rede; não é ideal para identificação de ataques complexos; se a rede possui muitos segmentos serão necessários muitos sensores. Como exemplos de NIDS podemos citar o *Snort*[4], o *Shadow*[5] e o *NetProwler*[6].

2.3 Virtualização

Uma das definições mais utilizadas atualmente para máquina virtual é *"uma cópia isolada e eficiente de uma máquina real"* (POPEK and GOLDBERG, 1974). Em outras palavras a virtualização é uma tecnologia que oferece uma camada de abstração dos verdadeiros recursos de uma máquina, provendo um *hardware* virtual para cada sistema (UHLIG, 2005). Utilizando a virtualização, uma única máquina física pode hospedar diversas máquinas virtuais (Virtual Machine (VM)) isoladas.

O conceito de virtualização surgiu há bastante tempo, para solucionar o problema de subutilização dos recursos computacionais dos *mainframes*. Com o passsar dos anos e o aumento da capacidade de processamento dos computadores pessoais a virtualização também passou a ser utilizada em computadores de pequeno porte.

As grandes vantagens do uso da virtualização em ambientes computacionais são:

[4] `http://www.snort.org`
[5] `http://www.open-mag.com/features/Vol_18/shadow/SHADOW.htm`
[6] `http://www.trtec.com.br/html/netprowler.html`

- Maior e melhor utilização dos recursos de *hardware*;
- Menor custo de implementação de servidores virtuais do que servidores reais;
- Facilidade de realização de *backups*;
- Otimização de espaço físico;
- Economia de energia.

2.3.1 O monitor de máquinas virtuais

O monitor de máquinas virtuais (Virtual Machine Monitor (VMM)) é a aplicação que cria a camada entre o *hardware* e as VMs. Segundo (LAUREANO, 2003), *"As finalidades primárias de um sistema operacional são habilitar aplicações a interagir com um hardware de computador e gerenciar recursos de hardware e software de um sistema. Por tal motivo, o monitor de máquinas virtuais pode ser definido como um sistema operacional para sistemas operacionais"*.

É o VMM que implementa e gerencia as VMs, interpretando e emulando o conjunto de instruções entre as máquinas virtuais e a máquina real (*hardware*). Entre as principais funções do VMM podemos citar: definição do ambiente de VMs; alteração do modo de execução do Sistema Operacional (SO) convidado; emulação das instruções e escalonamento do uso do processador para as VMs; gerenciamento do acesso a memória e disco das VMs; controle de acesso a dispositivos como disquetes, *interfaces* de rede e dispositivos Universal Serial Bus (USB). Um VMM deve apresentar três características básicas (POPEK and GOLDBERG, 1974):

- Eficiência: o VMM não deve intervir demais no tratamento das instruções. Somente as instruções que não puderem ser tratadas pelo Central Processing Unit (CPU) real é que devem ser tratadas pelo VMM;
- Integridade: todas as requisições aos recursos de *hardware* devem ser alocadas explicitamente pelo VMM (memória, processamento etc);
- Equivalência: o comportamento de execução do VMM deve ser semelhante ao da máquina real, ou seja, a VM não deve perceber a existência do VMM.

2.3.2 Tipos de virtualização

Existem basicamente duas maneiras de implementar o VMM em um sistema: a virtualização total e a paravirtualização. A virtualização total fornece ao SO visitante uma réplica do *hardware* real. Desta maneira não são necessárias modificações no SO visitante sendo executado sobre o VMM. Tal comportamento traz alguns inconvenientes, como o VMM ter de suportar um número muito elevado de dispositivos. A fim de tentar diminuir o impacto deste problema, a maioria das implementações de virtualização total utilizam dispositivos genéricos, que funcionam de maneira adequada com a maior parte dos dispositivos disponíveis (MATTOS et al., 2008). A Figura 3 mostra a implementação da virtualização total. Nesta figura é possível notar que o VMM é executado sobre o SO hospedeiro como qualquer outro aplicativo.

As ferramentas *open source* mais conhecidas para implementação da virtualização total são o *VirtualBox*[7] e o *KVM*[8].

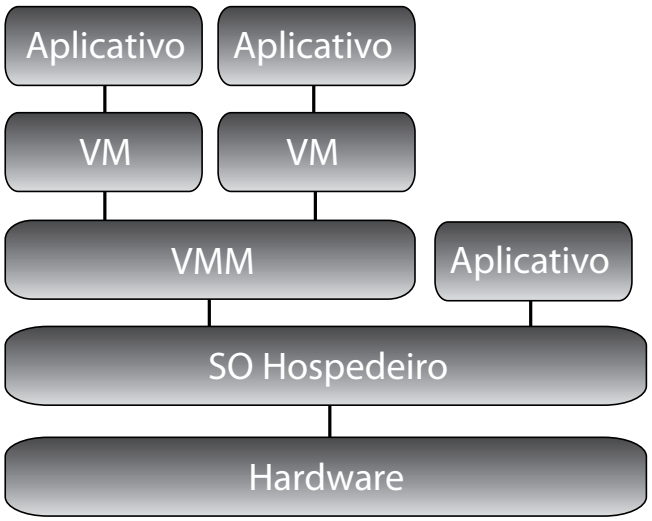

Figura 3: Implementação da virtualização total

Na paravirtualização o SO é modificado para requisitar o VMM toda vez que executar instruções sensíveis, ou seja, que possam alterar o estado do sistema. Isso elimina a necessidade de o VMM ter de testar todas as instruções, o que gera aumento no desempenho. Outro aspecto da paravirtualização é que os *drivers* da própria VM acessam os dispositivos de *hardware*, eliminando assim a necessidade da utilização de *drivers* genéricos (MATTOS et al., 2008). A Figura 4 mostra a implementação da paravirtualização. Nesta figura percebe-se que o próprio VMM atua como SO hospedeiro, diretamente sobre o hardware.

[7] `http://www.virtualbox.org`
[8] `http://www.linux-kvm.org`

Como exemplo de *software* podemos citar o *Xen*[9].

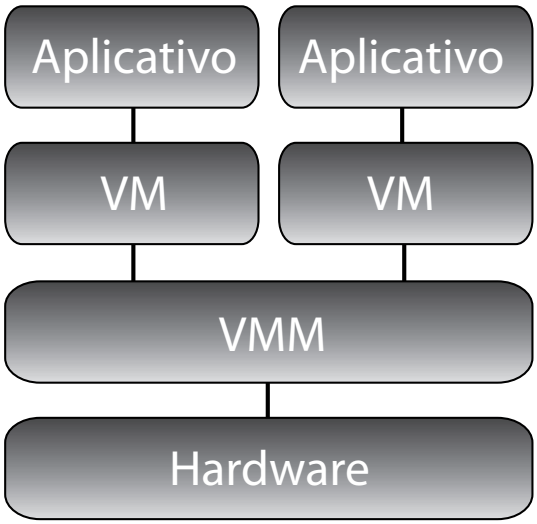

Figura 4: Implementação da paravirtualização

Hoje em dia, a presença de instruções de virtualização nos processadores *AMD* e *Intel* têm diminuído a disparidade de desempenho entre a paravirtualização e a virtualização total (UHLIG, 2005). Tal tecnologia possibilita que um único processador simule diversos processadores virtuais. É importante notar que esta tecnologia funciona diferente do que a utilização de diversos núcleos em um processador físico. Quando um processador tem dois núcleos, por exemplo, os núcleos trabalham em conjunto, dividindo as tarefas realizadas. Na tecnologia de virtualização de processadores, dois processadores virtuais trabalham de maneira independente, cada um atendendo sua determinada VM, por exemplo.

2.3.3 Ferramentas de virtualização

Esta seção apresenta um resumo do funcionamento de dois *softwares* de virtualização, o *VirtualBox* (virtualização total) e o *Xen* (paravirtualização).

2.3.3.1 *VirtualBox*

O *VirtualBox* é um *software* para virtualização que utiliza a abordagem de virtualização total. Por isso, quando utilizado com processadores sem suporte a tecnologia de virtualização, ainda perde em desempenho para *softwares* que implementam a paravirtualização, pois as máquinas virtuais não conseguem acessar diretamente o *hardware* real. Porém uma das principais vantagens da utilização do *VirtualBox* é a facilidade de

[9] `http://www.xen.org`

implementação. O *VirtualBox* é instalado diretamente no SO hospedeiro e é executado com um aplicativo qualquer. Dentro desta aplicação são criadas as VMs. Cada VM pode utilizar um sistema operacional distinto, que são conhecidos como SOs visitantes.

Todas as VMs atuam de maneira independente, podendo ser ligadas, desligadas, criadas e excluídas sem interferir umas nas outras. Além disso, o *VirtualBox* oferece diferentes recursos para configuração das rede entre as VMs. Desta forma é possível configurá-las para atuar de maneira independente ou em rede.

Além disso, o *VirtualBox* implementa uma ferramenta chamada *VBoxMachine*. Está ferramenta possibilita que o *VirtualBox* e as VMs sejam totalmente controlados através da linha de comando. Esse recurso possibilita a instalação do *VirtualBox* em servidores sem interface gráfica.

2.3.3.2 *Xen*

O *Xen* é um dos *softwares* mais populares de paravirtualização. A maior vantagem na utilização do *Xen* é que ele apresenta um desempenho melhor como VMM do que os *softwares* de virtualização total (BARHAM, 2003), quando o CPU da máquina real não oferece suporte a instruções de virtualização. No entanto, é necessário que o SO visitante seja portado pelo *Xen*. Isso não chega a ser um problema, pois atualmente o *Xen* suporta os principais SOs do mercado, como *Linux*, *FreeBSD* e *Windows*.

Para entender melhor a implementação da paravirtualização com o *Xen* é necessário entender primeiramente dois conceitos: o de *hypervisor* e o de domínio. Os domínios são as VMs do *Xen*. Estes domínios podem ser privilegiados (domínio 0) e não-privilegiados (domínio U). O *hypervisor* tem a função de controlar todos os recursos de comunicação, processamento e memória das VMs, porém não possui drivers para manipular estes dispositivos diretamente.

Quando a máquina real é ligada, uma VM do domínio 0 é criada. Este domínio é responsável por executar todas as aplicações de gerenciamento. Só é possível criar, ligar e desligar as VMs dos domínios U através do domínio 0. Na VM do domínio 0, é executado um SO com núcleo modificado que acessa todos os recursos da máquina real e comunica-se com as outras VMs dos domínios U. O núcleo deste SO precisa ser modificado para possuir os *drivers* dos dispositivos da máquina real e mais dois drivers para tratar as requisições de acesso realizadas pelas VMs do domínio U. Resumindo, somente a VM do domínio 0 pode acessar os recursos da máquina real, enquanto as VMs do domínio U

somente acessam estes recursos através do domínio 0.

Concluindo, o *Xen* apresenta recursos importantes para melhorar o desempenho do sistema, porém a configuração deste tipo de virtualização é mais complexa do que a virtualização total.

2.4 Diversidade de *software*

A diversidade de *software* é um mecanismo amplamente conhecido de tolerância a falhas de *software* (BESSANI et al., 2008). Este mecanismo baseia-se no conceito de que *softwares* distintos apresentam vulnerabilidades distintas (AVIZIENIS, 1984). Assim, mesmo que uma vulnerabilidade do sistema seja descoberta e explorada, os outros dispositivos do sistema, que apresentam implementação diferente, não apresentarão a mesma vulnerabilidade e continuarão seguros.

Porém é necessário pesar os prós e contras antes de decidir a quantidade de *softwares* a utilizar. Um aspecto importânte a se observar é que quanto maior o número de *softwares*, maior o custo de manutenção do sistema e maior a complexidade de administração do mesmo.

3 Um sistema tolerante a intrusão

Este capítulo apresenta uma descrição sobre a arquitetura e funcionamento do sistema proposto. Como visto na Seção 1.2 o principal objetivo deste sistema é ser tolerante a intrusões, baseando-se em técnicas de virtualização e diversidade de *software*. Isto é, o sistema deve ser capaz de permanecer seguro e estável mesmo após sofrer uma invasão. Para alcançar esse objetivo o sistema desenvolvido faz o monitoramento de servidores virtuais e, ao identificar uma intrusão ou tentativa de intrusão, o sistema atua de maneira reativa, desativando o servidor ameaçado, ativando um servidor substituto e reconfigurando o ambiente. Assim busca-se garantir que os danos ao servidor invadido sejam minimizados o máximo possível e que o ambiente como um todo continue funcionando de maneira segura.

Para que o sistema funcione de maneira adequada foram utilizados *softwares* de apoio. Um deles é um *software* IDS, que faz o monitoramento dos servidores e dispara alertas quando identifica algum evento suspeito. Além disso um *software* de virtualização também foi utilizado. Desta forma o sistema é basicamente composto por uma máquina real com o SO hospedeiro, pelo *software* para implementar virtualização, pelo *software* IDS, pelas VMs, por uma base de dados e por fim pelo sistema reativo, chamado de SDIR. Abaixo é apresentada uma breve descrição de cada um destes componentes.

- Máquina real – a máquina real ou hospedeira serve como base para todo o sistema. Essa máquina desempenha duas funções essenciais para funcionamento do sistema: o compartilhamento dos recursos de *hardware* com as máquinas virtuais e a execução do SO hospedeiro. Para que a máquina real consiga executar tais tarefas é importante que esta tenha bons recursos de *hardware*.

- SO hospedeiro – é o SO instalado na máquina real. A principal função do SO hospedeiro neste sistema é fazer a interface entre o *hardware* da máquina real e o *software* de virtualização.

- Software de virtualização – É o *software* utilizado para criar e gerenciar todas as VMs. Sua principal função é fazer a emulação do *hardware* para as VMs.

- VM – é cada uma dos sistemas instalados no *software* de virtualização. Cada uma das VMs funciona de maneira similar a uma máquina real. A sua função principal é implementar os servidores de fato. Nas VMs são instalados o SO convidado e os serviços disponíveis, tais como WEB, email, etc.

- IDS – o IDS é o *software* responsável por identificar os eventos maliciosos e executar ações de reação a esses eventos. Os IDS instalados em cada uma das VMs fazem o monitoramento dos eventos enquanto o IDS instalado no SO hospedeiro gerencia todos os IDSs instalados nas VMs. É também o IDS instalado no hospedeiro que executa SDIR.

- Base de dados - a base que contém as informações de todos os servidores virtuais. É com auxílio desta base que o SDIR executa todas as ações de recuperação do sistema.

- SDIR - o SDIR é o principal componente do sistema. É ele que executa todas as ações de recuperação do sistema.

A Figura 5 mostra a interação entre esses componentes.

O sistema foi desenvolvido para funcionar corretamente para até N-1 invasões para N servidores implementados. Cada um destes servidores está instalado em uma VM distinta. Podem ser utilizadas tantas VMs quanto os recursos de processamento da máquina real suportarem. O sistema pode ser basicamente configurado de duas maneiras: com todos os servidores virtuais ativos ao mesmo tempo ou deixando ativos somente os servidores virtuais que estão sendo utilizados no momento, dependendo das características do ambiente computacional que o qual será instalado.

A diferença básica destes dois tipos de cenários, é que quando todas as VMs estão ligadas, o tempo de recuperação do sistema é muito menor, já que não é necessário aguardar até que outro servidor virtual inicie. Porém esta configuração consome mais recursos computacionais e deve ser utilizada somente com máquinas com *hardware* suficiente. Já quando o sistema mantém ativas somente as VMs que estão sendo utilizadas, o consumo de recursos é bem menor. Esta configuração é indicada quando a máquina real tem poucos recursos de *hardware*.

Independentemente da configuração realizada, cada VM funciona como um servidor virtual, disponibilizando os serviços diversos, como por exemplo, WEB, email, SSH, etc. Em cada uma das máquina virtuais existe um HIDS que faz o monitoramento do tráfego nos servidores. Este *software* é responsável por disparar alertas para o servidor HIDS toda vez que identifica uma ação suspeita. Para gerenciar os HIDS instalados nos servidores virtuais, é utilizado um servidor HIDS, instalado no sistema operacional hospedeiro. Este servidor monitora de maneira constante todos os HIDS dos servidores virtuais, e recebe os alertas por eles enviados. Toda esta comunicação entre os agentes HIDS e o servidor HIDS é feita de maneira segura, atráves de uma chave criptográfica compartilhada.

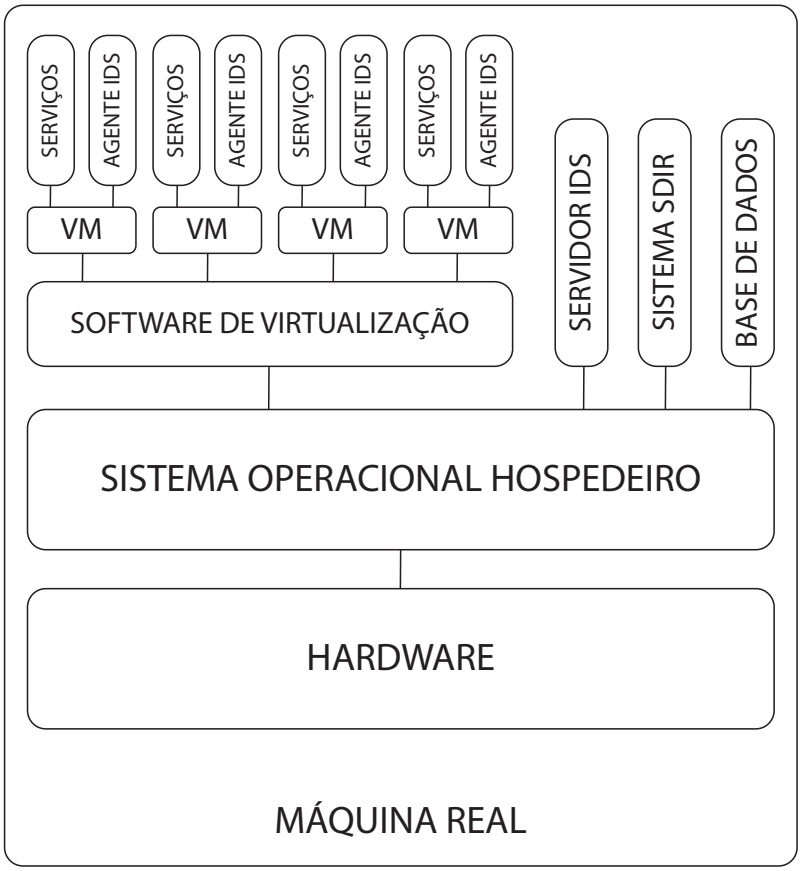

Figura 5: Sistema proposto

Desta forma, quando o agente HIDS identifica em evento suspeito, um alerta é gerado e envidado ao servidor HIDS de maneira criptografada. Após receber o alerta o servidor deve verificar se tal alerta refere-se a uma ação de invasão. Se esta ação for identificada como um evento normal, o servidor HIDS somente registra em *log* este alerta. Esta ação é executada para que tenha-se um histórico de todos os alertas disparados. Porém se o servidor HIDS identifica o alerta recebido como uma ação maliciosa, o mesmo executa o sistema de reação, o SDIR.

O SDIR é o sistema responsável por executar todas as ações de recuperação do sistema. Quando o SDIR é executado, ele recebe algunas informações sobre a invasão. Estas informações são enviadas pelo servidor HIDS. Porém estas informações não são suficientes para executar as ações necessárias de recuperação do sistema. Desta forma o SDIR utiliza uma base de dados auxiliar que contém informações detalhadas sobre cada um dos servidores virtuais implementados.

Com base nas informações recebidas do servidor HIDS, o SDIR consulta a base de dados para conseguir mais informações sobre o servidor atacado. É através destas informações que o SDIR executa a primeira ação de recuperação que consiste em desligar o servidor invadido. Tal ação é executada através de uma solicitação do sistema ao monitor de máquinas virtuais para que desligue imediatamente a VM comprometida. Após a execução desta ação a base de dados e atualizada, tornando a VM atacada indisponível ao sistema.

Em seguida, o próxima ação de recuperação executada pelo sistema é a escolha do servidor virtual que será ativado para substituir o servidor invadido. É neste momento que o mecanismo de diversidade de *software* é aplicado. Novamente com base nas informações consultadas da base de dados, o SDIR compara e escolhe uma VM similar porém com implementação diferente da VM atacada. Tal ação busca eliminar o risco de que um ataque explorando a mesma vulnerabilidade seja realizado no servidor substituto. Uma vez que o sistema escolhe a VM substituta, o SDIR ativa este novo servidor. Esta ativação pode acontecer de duas maneiras. Se o sistema estiver configurado para manter ativados somente os servidores que estão sendo utilizados no momento, então o SDIR solicita ao monitor de máquina virtuais que ative a VM substituta. Caso o sistema esteja configurado para manter todos os servidores ativos, mesmo os que não estão sendo utilizados, então esta etapa não é necessária.

A seguir o SDIR faz novamente a atualização da base de dados com as novas informações, para que o sistema identifique que o servidor que antes estava desativado, está agora em uso. Desta forma evita-se que o sistema tente ativar novamente esta VM em caso de nova invasão.

A última ação de recuperação executada pelo sistema é a reconfiguração da rede. Esta reconfiguração é feita para que o usuário continue acessando o serviço no mesmo endereço que estava acessando antes da invasão.

É importante ressaltar que uma vez que o sistema sofre N invasões (onde N é o número de VMs disponíveis) o mesmo deixa de funcionar por não existir VMs para substituir o

sistema invadido. Desta forma sempre que um servidor é invadido, é importante proceder com a recuperação do mesmo o mais rápido possível para que este esteja disponível novamente ao sistema.

4 *Implementação do protótipo*

O presente capítulo apresenta a implementação do protótipo do sistema proposto. Para facilitar o entendimento, a implementação do sistema será mostrada em etapas. Desta forma as próximas serão são organizadas da seguinte forma: a Seção 4.1 apresenta as configurações básicas do sistema, a Seção 4.2 apresenta a implementação das VMs utilizadas, a seção 4.3 apresenta a configuração do software IDS, a Seção 4.4 mostra a implementação do SDIR e por fim a Seção 4.5 apresenta a implementação da base de dados auxiliar;

4.1 Configurações básicas do sistema

Esta seção apresenta as configurações básicas necessárias para implementação do sistema tolerante a intrusão proposto. O ponto de partida para implementação é a escolha da máquina real do sistema. Para que o sistema funciona de maneira adequada é necessário que esta máquina tenha bons recursos de *hardware*.

A máquina deve possuir SO Linux e o *VirtualBox* como *software* de virtualização. A escolha pelo *VirtualBox* deve-se a dois fatores: é um *software* que permite o controle das VMs através de linhas de comando; é um *software* simples, o que reduz os riscos de má configuração do sistema (ver seção 1.2). A máquina real também deve possuir o *software* *OSSEC* HIDS instalado no modo *"server"*. O modo *server* é a configuração feita para que o HIDS atue como gerenciador de todos os outros agentes. Neste caso o *OSSEC* HIDS foi escolhido, pois tem uma arquitetura multi-plataforma, necessária para funcionamento do sistema, e fornece suporte a resposta ativa.

4.2 Implementação das VMs

Esta seção apresenta a implementação das VMs do sistema, bem como a configuração necessária para um desempenho adequado do servidor virtual. O primeiro passo é a criação das VMs. Podem ser criadas tantas VMs quanto a máquina real suportar. As VMs criadas podem ter qualquer sistema operacional que suporte o *software OSSEC* HIDS. O *software OSSEC* é quem faz o monitoramento das VMs.

Diferentemente da instalação na máquina real, nas VMs o *OSSEC* precisa estar instalado no modo *agent* e não no modo *server*. O modo *agent* é utilizado para que o *software* monitore a máquina na qual está instalado e envie alertas a um servidor pré-configurado. Além disso é necessário que as VMs sejam configuradas para utilizar a placa de rede em modo *hostonly*. Essa configuração faz com que o *VirtualBox* crie uma placa de rede virtual para o VMMs que serve como ponte para as VMs. Desta forma todas as máquinas virtuais criadas conseguem comunicar-se entre si e com a máquina real. Se esta configuração não for realizada os agentes *OSSEC* não conseguem comunicar-se com o servidor *OSSEC* e o sistema não funciona.

4.3 Configuração do *OSSEC* HIDS

Esta seção apresenta a configuração do *OSSEC* HIDS necessária para que o sistema funcione como proposto. A Subseção 4.3.1 apresenta a configuração dos agentes *OSSEC* e a Subseção 4.3.2 apresenta a configuração do servidor *OSSEC*.

4.3.1 Configuração dos agentes *OSSEC*

O primeiro passo na configuração do *OSSEC* é adicionar os agentes *OSSEC* (instalados nas VMs) no servidor *OSSEC* (instalado no SO hospedeiro). Essa configuração é necessária para que o servidor *OSSEC* identifique estes agentes e consiga gerenciar todos os alertas por eles emitidos. Para adicionar os agentes *OSSEC* deve-se utilizar a ferramenta *Manage Agents*. Esta é uma ferramenta do *OSSEC* que possibilita fazer o gerenciamento de todos os agentes *OSSEC* do sistema. Deve-se adicionar ao servidor *OSSEC* todos os agentes criados nas VMs.

Após adicionar todos os agentes, é necessário fazer a extração da chave de segurança de cada um dos agentes. Toda a comunicação entre os agentes *OSSEC* e o servidor *OSSEC* é realizada de maneira criptografada, para garantir que nenhum componente, que

não os agentes, possa manipular o servidor *OSSEC*. Para isto o *OSSEC* utiliza chaves independentes para cada um dos agentes do sistema.

Para finalizar a configuração dos agentes *OSSEC*, é necessário inserir a chave extraída no servidor em cada um dos agentes. Para isto utiliza-se novamente a ferramenta *Manage Agents*. Desta forma os agentes estão configurados para monitorar e enviar os alertas ao servidor *OSSEC*.

4.3.2 Configuração do servidor *OSSEC*

Esta subseção apresenta a implementação de um recurso fundamental para o funcionamento do sistema como proposto, a resposta ativa no servidor *OSSEC*. É este recurso que possibilita construir um sistema reativo, que execute uma ação com base no tipo de alerta recebido. Para isso é necessário configurar o *OSSEC* para utilizar a resposta ativa e configurar também o comando que será executado caso um alerta seja recebido. Para ambas as configurações precisa-se editar o arquivo de configuração do *OSSEC*, o `ossec.conf`. Para habilitar a resposta ativa no servidor *OSSEC* adiciona-se a estrutura mostrada na Figura 6 no arquivo `ossec.conf`:

```
1 <active-response>
2     <command>nome_do_comando</command>
3     <location>local</location>
4     <rules_id>regras</rules_id>
5 </active-response>
```

Figura 6: Estrutura utilizada para habilitar a resposta ativa no arquivo `ossec.conf`

Esta estrutura é composta basicamente por três parâmetros:

- *command* – deve-se informar o nome do comando que será executado;
- *location* – local onde o comando será executado. Utiliza-se a opção *"server"* pois o comando deverá ser executado no servidor *OSSEC*;
- *rules id* – deve-se informar o número das regras que executarão o comando. Pode-se utilizar mais de uma regra, neste caso estas devem estar separadas por vírgula.

Agora é necessário criar o comando que será executado pelo servidor *OSSEC*. Essa configuração é feita também no arquivo de configuração do *OSSEC*, o `ossec.conf`. A estrutura que deve-se utilizar é mostrada na Figura 7:

```
<command>
    <name>nome_do_comando</name>
    <executable>arquivo_executavel</executable>
    <expect></expect>
    <timeout_allowed>no</timeout_allowed>
</command>
```

Figura 7: Estrutura utilizada para criar um comando no arquivo `ossec.conf`

Esta estrutura é composta por quatro parâmetros:

- *name* – nome do arquivo que está sendo criado. Deve ter o mesmo nome que foi utilizado no parâmetro *"command"* da etapa anterior;
- *executable* – nome do arquivo que será executado.
- *expect* – parâmetros esperado pelo arquivo executável. Deve-se deixar esse campo vazio;
- *timeout allowed* – se o comando executado terá tempo para expirar. Deve-se deixar esse campo vazio.

Com estes parâmetros o servidor *OSSEC* é configurado para executar um comando sempre que receber um alerta dos agentes *OSSEC* com determinadas regras.

4.4 Implementação do SDIR

Esta seção apresenta de maneira detalhada a implementação do SDIR. O SDIR é o sistema responsável por executar todas as ações de reconfiguração após um ataque. Este sistema foi implementado através de um arquivo *Shell Script*. Toda vez que o servidor *OSSEC* recebe um alerta e o reconhece como um evento malicioso o SDIR é executado.

A reconfiguração do sistema consiste na execução de diversas ações para garantir que o sistema continue provendo o serviço de maneira confiável e segura. A seguir é apresentado o arquivo *shell script* utilizado para implementar o SDIR.

```
#!/bin/sh

# DECLARACAO DAS VARIAVEIS
AGENTNAME=$6
RULEID=$5
```

```
6  BD="/var/ossec/etc/clientconf"
7  STS="off"
8
9  # GRAVAR INFORMACOES NO LOG
10 echo "`date` $0 $1 $2 $3 $4 $5 $6 $7" >> /var/ossec/logs/active-responses.log
11
12 # CONSULTAR INFORMACOES SOBRE A VM INVADIDA
13 LINHA=`grep "$AGENTNAME" "$BD"`
14 VM=`echo "$LINHA" | awk -F";" '{print $1}'`
15 IP=`echo "$LINHA" | awk -F";" '{print $3}'`
16 OS=`echo "$LINHA" | awk -F";" '{print $4}'`
17 SERV=`echo "$LINHA" | awk -F";" '{print $5}'`
18 HTTP=`echo "$SERV" | awk -F"," '{print $1}'`
19 SMTP=`echo "$SERV" | awk -F"," '{print $2}'`
20 FTP=`echo "$SERV" | awk -F"," '{print $3}'`
21 NAT=`echo "$LINHA" | awk -F";" '{print $7}'`
22 L1=`echo "$LINHA" | awk -F";" '{print $8}'`
23
24 # ATUALIZAR A BASE DE DADOS
25 sed -i ""$L1"s/on/infected/" $BD
26 sed -i ""$L1"s/"$NAT"/nenhum/" $BD
27
28 # REMOVER INFORMACOES DE NAT DA VM INVADIDA
29 iptables -t nat -D PREROUTING -p TCP -d "$NAT" --dport 80 -j DNAT --to "$IP"
30 iptables -D FORWARD -p TCP -d "$IP" --dport 80 -j ACCEPT
31
32 # DESLIGAR VM INVADIDA
33 vboxmanage controlvm "$VM" poweroff &
34
35 # VERIFICAR QUAL MELHOR VM PARA SUBSTITUIR DE ACORDO COM O ID DO ALERTA
36 case $RULEID in
37
38    5503)
39       IVMNOVA=`grep "$STS" $BD | grep -v "$OS" | head -1`
40    ;;
41    5902)
42       IVMNOVA=`grep "$STS" $BD | grep -v "$HTTP" | head -1`
43    ;;
44    5716)
45       IVMNOVA=`grep "$STS" $BD | grep -v "$SMTP" | head -1`
46    ;;
47          *)
48       echo "Nenhuma VM disponivel"
```

```
49    ;;
50 esac
51
52 # TESTA A VARIAVEL IVMNOVA
53 if  [ "x${IVMNOVA}" = "x" ]
54
55 # SE VERDADEIRA
56    then
57       echo "Nenhuma VM disponivel"
58
59 #SE FALSA
60    else
61       VM2=`echo "$IVMNOVA" | awk -F";" '{print $1}'`
62       IP2=`echo "$IVMNOVA" | awk -F";" '{print $3}'`
63       L2=`echo "$IVMNOVA" | awk -F";" '{print $8}'`
64
65       sed -i ""$L2"s/off/on/" $BD
66       sed -i ""$L2"s/nenhum/"$NAT"/" $BD
67       vboxmanage startvm "$VM2"
68       iptables -t nat -A PREROUTING -p TCP -d "$NAT" --dport 80 -j DNAT --to "
            $IP2"
69       iptables -A FORWARD -p TCP -d "$IP2" --dport 80 -j ACCEPT
70 fi
```

O arquivo SDIR faz uso de informações enviadas pelo servidor *OSSEC* no momento da intrusão. As informações utilizadas pelo SDIR são: nome do agente *OSSEC* que disparou o alerta e número da regra. Através do nome do agente que gerou o alerta descobre-se qual máquina virtual foi invadida. Já o número da regra identifica o tipo de ataque sofrido pela VM. Estas informações são essenciais para que o SDIR possa executar as ações de reconfiguração do sistema.

A primeira ação executada pelo SDIR é a documentação do alerta. Através de um arquivo de *log* chamado `active-responses.log` ele armazena todas as informações de sobre os ataques sofridos pelas VMs. Em seguida o SDIR faz o armazenamento de todas as informações referentes a VM invadida. Tal ação é executada consultando-se uma base de dados previamente criada (a implementação desta base de dados é apresentada na Seção 4.5). A consulta é feita com base no nome do agente *OSSEC* que gerou o alerta. Através desta consulta identifica-se o ID da VM atacada, bem como todas as informações referentes aos *softwares* instalados na VM e *Network Address Translator (NAT)*. Estas informações são armazenadas em uma variável para que possam ser utilizadas posteriormente. Uma

vez que o SDIR tem as informações sobre a VM atacada, a próxima ação executada é a atualização da base de dados e o desligamento da VM, para garantir que a mesma fique vulnerável o menor tempo possível. A atualização da base de dados é feita para mudar o *status* da VM invadida e torná-la indisponível ao sistema. Também é feita a atualização das informações de NAT da VM invadida. Já o desligamento é realizado utilizando-se a ferramenta *VBoxManage* juntamente com o nome da VM atacada. A última ação realizada para eliminar a VM invadida é a eliminação das regras de NAT através da utilização da ferramenta *Iptables*.

As próximas ações realizadas pelo SDIR são referentes a seleção da VM que será ligada para substituir a VM invadida. Para garantir que o mecanismo de diversidade de *software* funcione, o SDIR deve identificar se a vulnerabilidade explorada aconteceu no SO da VM ou em um serviço. Essa identificação é feita com o auxílio da outra informação recebida do servidor *OSSEC*: o número da regra. Se a vulnerabilidade explorada foi do SO, então o SDIR seleciona uma VM com SO diferente da VM invadida. Se a vulnerabilidade explorada foi de um serviço, então o SDIR seleciona uma VM com uma implementação diferente do mesmo serviço. Isso garante que a mesma vulnerabilidade não seja explorada na VM selecionada. A seleção da VM é feita consultando-se a base de dados e verificando-se as VMs disponíveis para utilização. Em seguida, dependendo do número da regra recebido do servidor *OSSEC*, o SDIR faz uma comparação dos *softwares* instalados na VM invadida e nas VMs disponíveis. Assim que uma VM disponível é identificada como tendo *software* diferente, esta é selecionada para substituir a VM invadida. A seguir todas as informações da VM selecionada são armazenadas para utilização posterior.

A próxima ação executada é novamente a atualização da base de dados. O status da VM selecionada é alterado para informar ao sistema que a mesma está sendo utilizada. As informações de NAT desta VM também são atualizadas. A seguir, novamente com o auxílio da ferramenta *VBoxManage* o SDIR inicia a VM selecionada. Por fim, o SDIR atribui o mesmo endereço de NAT do servidor, utilizando novamente o *Iptables*. Isso é feito para que a VM substituta possa ser acessada através do mesmo endereço Internet Protocol (IP) da VM invadida.

Todas estas ações em conjunto fazem com que o sistema seja reconfigurado, de acordo com o ataque sofrido, da melhor maneira possível. Estas ações também garantem que mesmo após uma invasão o sistema restabelecerá o estado de estável e seguro, ou seja, tolerante a intrusões.

4.5 Implementação da base de dados

Esta seção apresenta a implementação e a estrutura da base de dados utilizada pelo SDIR. Nesta base, encontram-se todas as informações que o SDIR necessita para executar suas ações de maneira adequada. Esta base de dados poderia ser implementada utilizando-se qualquer banco de dados, porém optou-se por utilizar um arquivo CSV (*Comma Separated Values*) por ser fácil de manipular com comandos simples como *grep*, *cut*, *sed* e *awk*. A Figura 8 mostra um exemplo da base de dados utilizada no sistema.

```
1 vm1;(s1);10.10.10.101;debian;apache;postfix;proftpd;on;192.168.10.201;1
2 vm2;(s2);10.10.10.102;debian;apache;xmail;wuftpd;infected;nenhum;2
3 vm3;(s3);10.10.10.103;ubuntu;apache;postfix;wuftpd;on;192.168.10.202;3
4 vm4;(s4);10.10.10.104;ubuntu;cherokee;xmail;proftpd;off;nenhum;4
```

Figura 8: Exemplo de arquivo utilizado como base de dados

A seguir apresentamos todas as infomações contidas na base de dados, bem como sua função.

Todas as informações apresentadas na Figura 8 tem uma função específica. Abaixo é apresentado a descrição de cada uma destas informações, bem como sua função:

- Nome do agente *OSSEC* – o nome do agente *OSSEC* é a informação que o arquivo `invasão.sh` recebe do servidor *OSSEC*. É através desta informação que a VM invadida é identificada.
- Nome da VM – esta informação é utilizada em conjunto com a ferramenta *VBoxManage* para efetuar o ligamento e o desligamento das VMs.
- IP da VM – esta informação é utilizada para fazer a configuração de NAT no sistema.
- SO da VM – esta informação é utilizada para fazer a seleção da VM adequada, caso o sistema tenha sofrido um ataque que explorou vulnerabilidades do SO.
- *Softwares* utilizados para implementar os serviços na VM - é utilizada para fazer a seleção da VM adequada, caso o sistema tenha sofrido um ataque que explorou vulnerabilidades de um dos serviços providos pela VM.
- *Status* da VM – esta informação identifica o status atual da VM. As VMs podem apresentar os seguintes status:
  - *on* – a VM está ligada;

  - *off* – a VM está desligada e disponível;
  - *infected* – a VM está desligada e indisponível por conta de uma invasão.

- *NAT* – está é o endereço de *NAT* utilizado na VM. Quando a VM esta com status *"off"* ou *"infected"* este campo apresenta a informação "nenhum", pois nenhum endereço de *NAT* está sendo utilizado.

- Número da linha – essa informação é utilizada pelo SDIR para facilitar a atualização da base de dados.

5 *Experimentos e resultados*

Esta seção apresenta todos os testes realizados para verificação do desempenho do sistema proposto. Através destes testes foi possível fazer uma medição do tempo utilizado pelo sistema para recuperação, em cenários distintos e avaliar a viabilidade de utilização do sistema em um ambiente corporativo.

A máquina utilizada para realização dos experimentos foi um *notebook Apple Macbook* com a seguinte configuração: processador *Intel Core 2 Duo* T8300 2.4 GHz e 3 Mb de memória cache L2, memória principal de 4 GB operando a freqüência de 800MHz, disco rígido de 320 GB 7200 Rotations Per Minute (RPM). O sistema operacional utilizado na máquina foi o *Ubuntu 9.10 (Karmic Koala 64 bits)*. Os servidores virtuais utilizados nos testes foram os mesmos descritos no Capítulo 4 .

Para fins de experimento o sistema foi configurado para reagir em três situações: falha na autenticação do *root*, criação de um usuário no sistema e falha na autenticação Secure Shell (SSH). Como visto no Capítulo 3, o sistema é ativado através do recebimento de uma mensagem de alerta do servidor virtual. Dessa forma é importante ressaltar que tais ações foram escolhidas por serem fáceis de simular. Para que fosse possível verificar se o sistema faria a escolha da máquina virtual com base no tipo de ataque baseado em técnicas de diversidade de *software*, foi definido o seguinte:

- Se o alerta fosse gerado por falha na autenticação do *root*, o sistema deveria escolher uma nova máquina virtual com sistema operacional diferente (simulando um alerta gerado por falha no sistema operacional);
- Se o alerta fosse gerado por criação de usuário no sistema, o sistema deveria escolher uma nova máquina virtual com servidor Hyper Text Transfer Protocol (HTTP) diferente (simulando um alerta gerado por falha no servidor HTTP);
- Se o alerta fosse gerado por falha na autenticação do SSH o sistema deveria escolher uma nova máquina virtual com servidor Simple Mail Transfer Protocol (SMTP)

diferente (simulando um alerta gerado por falha no servidor SMTP);

A Figura 9 mostra o cenário simulados nos testes.

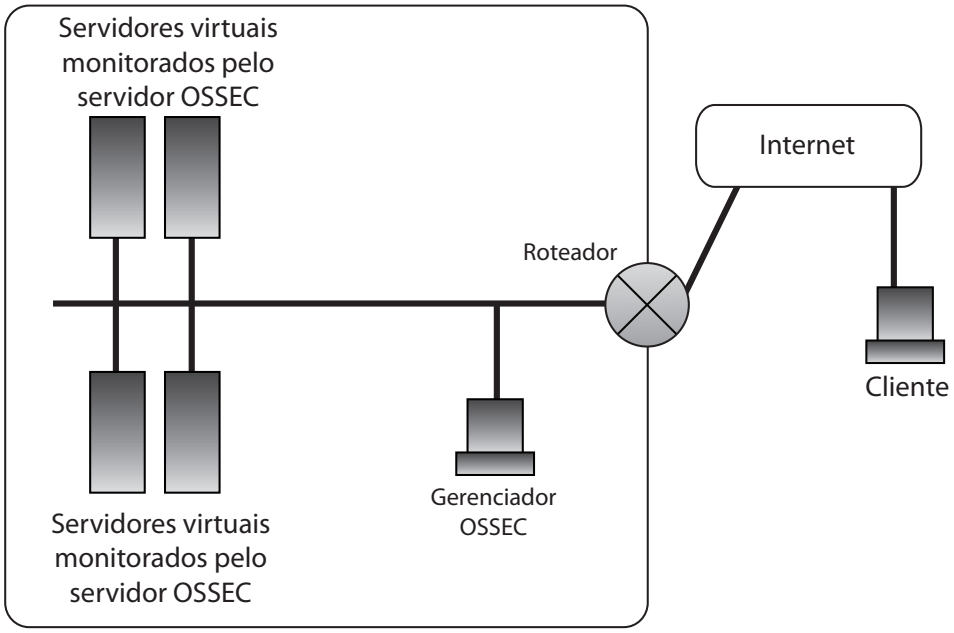

Figura 9: Cenário simulado nos experimentos

5.1 Experimetos

Esta seção descreve os experimentos realizados no sistema em dois cenários distintos. O primeiro cenário apenas as os servidores virtuais que estão sendo utilizados no momento permanecem ligados. Os servidores de *backup* ficam desligados e somente são ativados caso o sistema sofra um ataque. No segundo cenário, todos os servidores virtuais permaneceram ligados, assim quando o sistema sofre um ataque, não é necessário aguardar que a VM seja ligada, bastando que as configurações de NAT sejam alteradas.

No primeiro experimento dois servidores virtuais foram ligados e dois servidores virtuais permaneceram desligados. Então um ataque foi simulado a um dos servidores. Imediatamente após a simulação do ataque, o agente OSSEC identificou esta ação e disparou um alerta ao servidor OSSEC. Este por sua vez identificou o alerta como um ataque e executou o SDIR. O SDIR fez todas as consultas necessárias, desligou o servidor virtual atacado e ativou um servidor virtual substituto. Todas as configurações de NAT também foram feitas.

No segundo experimento utilizou-se um cenário diferente do primeiro. Neste cenário todos os servidores virtuais permaneceram ligados durante todo o tempo, mesmo que

não estivessem sendo utilizados. Desta forma, assim como no primeiro experimento, um ataque foi simulado a um dos servidores que estavam provendo algum serviço no momento. Novamente após o ataque o agente IDS identificou a ação e disparou um alerta. O servidor OSSEC mais uma vez executou o SDIR que fez as consultas necessárias, desligou o servidor invadido e reconfigurou o *NAT* para uma das VMs ativas.

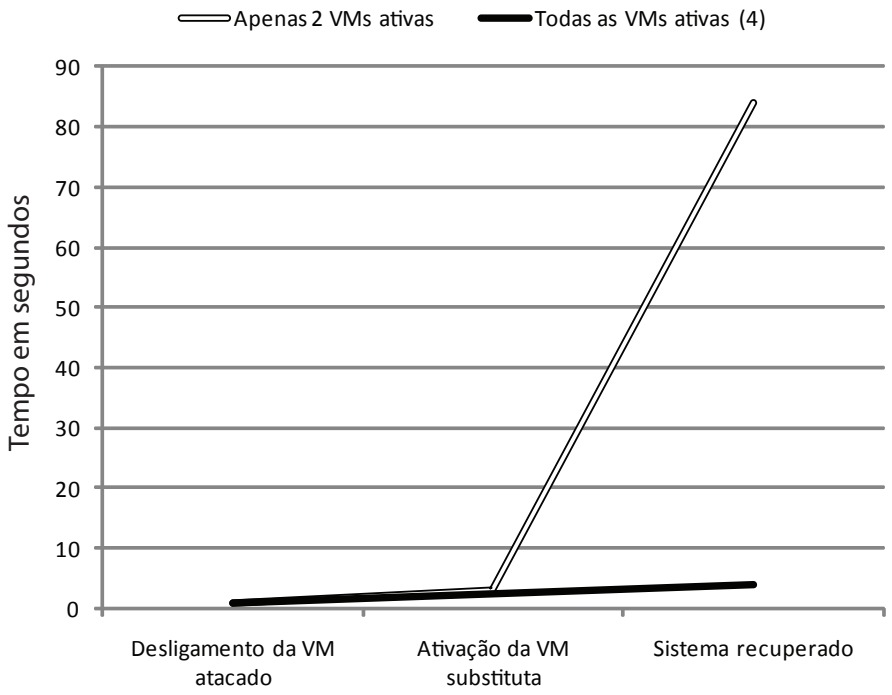

Figura 10: Desempenho do sistema nos dois experimentos

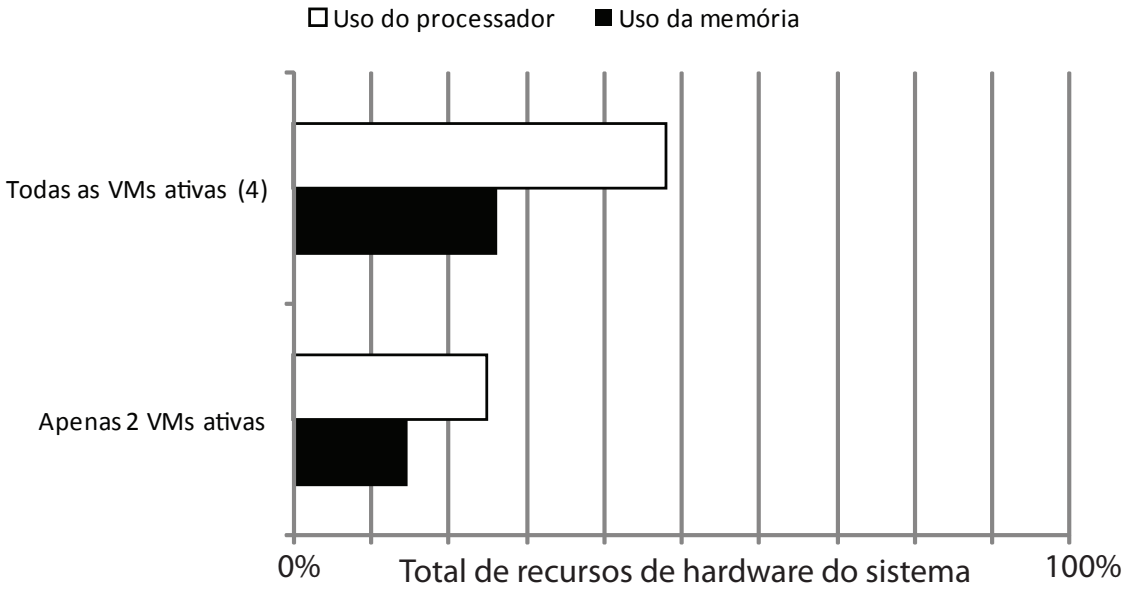

Figura 11: Desempenho do sistema nos dois experimentos

Ao final dos dois experimento verificou-se que o sistema se comportou da maneira esperada atingindo os objetivos de permanecer seguro e estável após uma invasão em ambas as situações. O mecanismo de diversidade de *software* também funcionou de maneira adequada em ambos os experimentos.

Na Figura 10 podemos observar que até o momento de escolher a VM substituta, os dois sistemas apresentam desempenho semelhante, porém a partir deste ponto até a recuperação total do sistema o segundo cenário mostrou-se muito mais rápido. Porém, podemos observar na Figura 11 que a quantidade de recursos consumidos quando todas as VMs do sistema estão ligadas é superior.

6 Conclusões

Este trabalho teve como objetivo apresentar a implementação de um sistema tolerante a intrusões baseado em técnicas de diversidade de software e virtualização. Desta forma foi possível desenvolver um sistema reativo a intrusões eficiente e simples. Para diminuir a complexidade do sistema, foram utilizados *softwares* para auxiliar o funcionamento do mesmo. Buscou-se utilizar softwares estáveis, usados amplamente, a fim de garantir o bom funcionamento do sistema como um todo.

A principal vantagem de utilizar tal sistema é tornar um servidor físico tolerante a falhas. Utilizando a tecnologia de virtualização foi possível replicar máquinas virtuais em um mesmo servidor físico, tornando possível a recuperação do serviço em caso de ataque. A técnica de diversidade de *software* foi também fundamental na elaboração deste sistema, tendo em visto que a utilização de *softwares* idênticos facilitaria um novo ataque – considerando que as vulnerabilidades de ambos os servidores seria a mesma.

O trabalho apresentado atingiu todos os objetivos estipulados em sua proposta inicial (apresentados na Seção 1.2). Através deste trabalho, foi possível implementar um sistema que ao detectar um ataque, analisa as informações do ataque e com base nessas informações age de maneira reativa, desativando o servidor atacado e iniciando um outro servidor similar. O sistema ainda faz automaticamente a reconfiguração da rede para que este processo seja transparente ao usuário. Também alcançou-se os objetivos de implementar um sistema simples e de custo reduzido – através da utilização de somente uma máquina real para replicação dos servidores virtuais.

6.1 Trabalhos futuros

Ao final do trabalho, mesmo tendo alcançado todos os objetivos, foram observados aspectos que podem ser trabalhados para melhorar o desempenho do sistema apresentados no Capítulo 5. Assim espera-se que o presente trabalho sirva como um motivador para novos estudos baseado em tolerância a intrusões através da utilização de diversi-

dade de *software* em sistemas virtualizados, e estimule o estudo de novas técnicas de implementação de ambientes computacionais seguros e de baixa complexidade.

Desta forma estão listados abaixo alguns tópicos para possíveis trabalhos futuros:

- Utilização de um sistema de virtualização com melhor desempenho como *KVM* a fim de aumentar o desempenho do sistema;
- Implementação de soluções para resolver o problema de não funcionamento do serviço caso todas as máquinas virtuais sejam invadidas;
- Implementação de novas ações de reação a ataques como identificação do invasor, bloqueio de acesso, etc;
- Comparação da abordagem utilizada nesta proposta com outras abordagens;
- Elaboração de experimentos mais complexos com intuito de fornecer resultados mais concretos relacionados ao sistema.

Lista de Abreviaturas

IDS Intrusion Detection System

VM Virtual Machine

AVI Atack - Vulnerability - Intrusion

HIDS Host Intrusion Detection System

NIDS Network Intrusion Detection System

VMM Virtual Machine Monitor

SO Sistema Operacional

USB Universal Serial Bus

CPU Central Processing Unit

HTTP Hyper Text Transfer Protocol

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

IP Internet Protocol

NAT Network Address Translator

RPM Rotations Per Minute

Referências

AMOROSO, E. G. **Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response**. Intrusion.Net Books, 1999.

AVIZIENIS, A.; KELLY, J. P. J. Fault Tolerance by Design Diversity: Concepts and Experiments. **Computer**, Los Alamitos, CA, USA, v.17, p.67–80, 1984.

AXELSSON, S. The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. In: IN PROCEEDINGS OF THE 6TH ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY, 1999. [s.n.], 1999. p.1–7.

AXELSSON, S. Intrusion Detection Systems: A Survey and Taxonomy. University of Technology Goteborg, 2000. Relatório técnico.

BARHAM, P. et al. Xen and the art of virtualization. In: SOSP '03: PROCEEDINGS OF THE NINETEENTH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 2003. **Proceedings...** New York, NY, USA: ACM, 2003. p.164–177.

BESSANI, A. N. et al. On the Effects of Diversity on Intrusion Tolerance. University of Lisbon, 2008. Relatório técnico.

CACHIN, C.; PORITZ, J. A. Hydra: Secure replication on the Internet. IBM Research, 2002. Relatório técnico.

CAREY, N.; CLARK, A.; MOHAY, G. **IDS Interoperability and Correlation Using IDMEF and Commodity Systems**, chapterInformation and Communications Security, p.252–264. Springer Berlin, 2002.

CASWELL, B.; BEALE, J.; (AUTHOR), A. B. **Snort IDS and IPS Toolkit**. Syngress, 2007.

CHANDRA, T. D.; TOUEG, S. Unreliable Failure Detectors for Reliable Distributed Systems. **Journal of the ACM**, [S.l.], v.43, p.225–267, 1996.

CORREIA, M.; VERíSSIMO, P.; NEVES, N. F. The architecture of a secure group communication system based on intrusion tolerance. In: IN INTERNATIONAL WORKSHOP ON APPLIED RELIABLE GROUP COMMUNICATION, 2001. [s.n.], 2001. p.17–22.

DEBAR, H.; DACIER, M.; WESPI, A. Towards a taxonomy of intrusion-detection systems. **Computer Network**, New York, NY, USA, v.31, n.9, p.805–822, 1999.

DENNING, D. E. An intrusion-detection model. **IEEE Transactions on software engineering**, [S.l.], v.13, n.2, p.222–232, 1987.

HEBERLEIN, T. et al. A Network Security Monitor. University of California, 1990. Relatório técnico.

JACKSON et al. A Phased Approach to Network Intrusion Detection. **14th National Computing Security Conference**, [S.l.], v.0, p.0, 1991.

KREIBICH, C. Policy-controlled Event Management for Distributed Intrusion Detection. In: IN PROCEEDINGS OF THE 4TH INTERNATIONAL WORKSHOP ON DISTRIBUTED EVENT BASED SYSTEMS, 2005. [s.n.], 2005.

LAUREANO, M.; E EDGARD JAMHOUR, C. M. Deteccao de intrusao em maquinas virtuais. **5 Simposio de Seguranca em Informatica**, [S.l.], v.0, p.0, 2003.

LITTLEWOOD, B.; STRIGINI, L. Redundancy and Diversity in Security. In: COMPUTER SECURITY ESORICS 2004, 9TH EUROPEAN SYMPOSIUM ON RESEARCH COMPUTER SECURITY, LNCS 3193, 2004. Springer, 2004. p.423–438.

LUNT, T. Detecting Intruders in Computer Systems. In: IN PROCEEDINGS OF THE 1993 CONFERENCE ON AUDITING AND COMPUTER TECHNOLOGY, 1993. [s.n.], 1993.

MATTOS, D. M. F. Virtualizacao: VMWare e Xen. Universidade Federal do Rio de Janeiro, 2008. Relatório técnico.

MELL, P. et al. **An Overview of Issues in Testing Intrusion Detection Systems**.

MICHAEL E. WHITMAN, H. J. M. **Principles of Information Secuirty**. Thomson Course Technology, 2007.

NEUMANN, P. G.; BARNES, A. **Practical Architectures for Survivable Systems and Networks: Phase-One Final Report**.

OBELHEIRO, R. et al. How Practical are Intrusion-Tolerant Distributed Systems? Universidade de Lisboa, 2006. Relatório técnico.

PAXSON, V. Bro: a system for detecting network intruders in real-time. In: SSYM'98: PROCEEDINGS OF THE 7TH CONFERENCE ON USENIX SECURITY SYMPOSIUM, 1998. **Proceedings...** Berkeley, CA, USA: USENIX Association, 1998. p.3–3.

POPEK, G. J.; GOLDBERG, R. P. Formal requirements for virtualizable third generation architectures. **Commun. ACM**, New York, NY, USA, v.17, n.7, p.412–421, 1974.

SEBRING, M. M.; WHITEHURST, R. A. **Expert Systems in Intrusion Detection: A Case Study**.

SMAHA, S. E. **Haystack: An Intrusion Detection System**.

TAN, K. M. C.; MAXION, R. A. The Effects of Algorithmic Diversity on Anomaly Detector Performance. In: DSN '05: PROCEEDINGS OF THE 2005 INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2005. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2005. p.216–225.

UHLIG, R. Intel virtualization technology. Intel, 2005. Relatório técnico.

ULVILA, J. W.; GAFFNEY, J. E. Evaluation of Intrusion Detection Systems. **Journal of Research of the National Institute of Standards and Technology**, [S.l.], v.108, p.453–473, 2003.

VACCARO, M.; LIEPINS, G. E. Detection of Anomalous Computer Session Activity. **Security and Privacy, IEEE Symposium on**, Los Alamitos, CA, USA, v.0, p.280, 1989.

VERíSSIMO, P. E.; NEVES, N. F.; CORREIA, M. P. Intrusion-tolerant architectures: Concepts and design. In: ARCHITECTING DEPENDABLE SYSTEMS, VOLUME 2677 OF LECTURE NOTES IN COMPUTER SCIENCE, 2003. Springer-Verlag, 2003. p.3–36.

WINKELER, J. A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks. **Proc. 13th National Computer Security Conference**, [S.l.], v.0, p.115–124, 1990.

WU, Y.-S. et al. Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS. **Computer Security Applications Conference, Annual**, Los Alamitos, CA, USA, v.0, p.234, 2003.